TEXAS A&M UNIVERSITY

CSCE 421: Machine Learning

Lecture 5

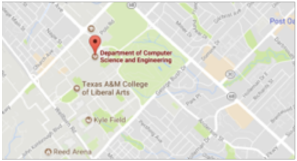
Overview

- Linear Regression: What have we learnt so far
    - Representation
    - Analytic Solution: Ordinary Least Squares
- Linear Regression: Numerical solution
    - General gradient descent
    - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression

Overview

- Linear Regression Basics
  - Representation
  - Analytic Solution: Ordinary Least Squares
- Linear Regression: Numerical solution
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression

Linear Regression: Example



Apartment Amenities

Pet Policy

Dogs Allowed: Breed Restrictions May Apply.

- $250 Fee
- 2 Pet Limit

Cats Allowed

- $250 Fee
- 2 Pet Limit

Fitness & Recreation

- Fitness Center
- Pool
- Volleyball Court

Living Space

- Carpet
- Attic
- Crown Molding
- Views
- Walk-In Closets

Outdoor Space

- Balcony
- Patio
- Porch
- Yard
- Barbecue Area
- Barbecue/Grill

Parking

Surface Lot

6 spaces; Assigned Parking.

Services

- Maintenance on site
- Property Manager on Site
- Bilingual
- Courtesy Patrol
- Trash Pickup - Curbside
- Recycling
- Planned Social Activities
- Pet Play Area

Kitchen

- Dishwasher
- Disposal
- Ice Maker
- Granite Countertops
- Kitchen
- Microwave
- Oven
- Refrigerator
- Freezer
- Instant Hot Water

Property Information

- Built in 2015
- 442 Units/2 Stories

Lease Length

August 1 - July 22

Outdoor Space

- Grill
- Waterfront
- Pond

Features

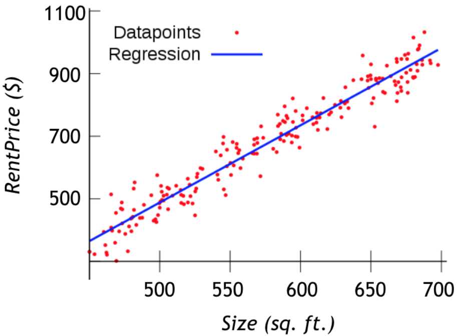
- High Speed Internet Access
- Washer/Dryer
- Air Conditioning
- Heating
- Ceiling Fans
- Smoke Free
- Cable Ready
- Tub/Shower
- Framed Mirrors

Source: apartments.com

$$RentPrice = w_0 + w_1 \times Size + w_2 \times DistanceFromCS + \ldots$$

Linear Regression: Example

$$RentPrice = w_0 + w_1 \times Size + w_2 \times DistanceFromCS + \ldots$$



Linear Regression: Example

- Input $\mathbf{x}$: apartment attributes (e.g., size, neighborhood)
- Output y: rent price of apartment
- Model parameters $\mathbf{w}$
- Linear model: $y = f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\mathbf{x} = w_1 x + 1 + \ldots + w_D x_D$
- Non-linear model: $y = f(\mathbf{x}|\mathbf{w}) = \mathbf{w}^T\phi(\mathbf{x})$

Linear Regression: Example

$$RentPrice = w_0 + w_1 \times Size + w_2 \times DistanceFromCS + \ldots$$

$$\mathrm{RentPrice} = w_0 + w_1 \times \mathrm{Size} + w_2 \times \mathrm{DistanceFromCS} + \ldots$$

How do we find the unknown model parameters $\{w_0, w_1, w_2, \ldots\}$?

We use training data!

Training Sample	Size (sq.ft.)	DistanceFromCS (miles)	RentPrice ($)
1	498	11.9	675
2	513	8.6	750
3	621	8.3	800
4	710	3.4	965
...	...	...	...

Linear Regression

- **Representation:** linear combination of features

  $f : \mathbf{x} \rightarrow y, \;\; f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- **Evaluation**: Minimizing the residual sum of squares

  $\min_{\mathbf{w}} RSS(\mathbf{w}), \;\; RSS(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$
- **Optimization**: Analytical $\rightarrow$ Ordinary least squares (OLS) solution

  $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

  Is this a global minimum? $\rightarrow$ 2nd derivative test

Linear Regression: Global minimum

Theorem

Consider an optimization problem

$$\min f(\mathbf{x}) \;\; \text{s.t.} \;\; \mathbf{x} \in \Omega$$

where f is a convex function and Ω is a convex set.

Then any **local** minimum is also a **global** minimum.

How do we find in f is convex?

Linear Regression: Global minimum

Definition 1

A set Ω is convex if for all $x, y \in \Omega$ and for all $\lambda \in [0,1]$

$$\lambda x + (1-\lambda)y \in \Omega$$

[In practice, if we draw a line between any two points of the set, every point on the line still lies within this set]

convex non-convex

Linear Regression: Global minimum

Definition 2

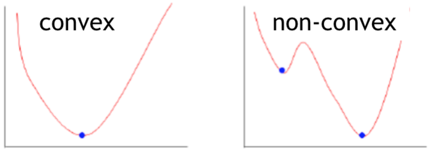
A function f is convex if its domain $\mathrm{dom}(f)$ is a convex set and for all $x, y \in \mathrm{dom}(f)$ and $\lambda \in [0,1]$

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

[In practice, the line segment connecting $(x, f(x))$ and $(y, f(y))$ should sit above the graph]

The second-derivative test

If the Hessian matrix $\mathbf{H_f}$ of f is positive semi-definite, then f is convex.
i.e. $\mathbf{u}^T \mathbf{H_f} \mathbf{u} \geq 0 \; , \;\; \forall \mathbf{u}$



Overview

- Linear Regression
  - Representation
  - Analytic Solution: Ordinary Least Squares
- Linear Regression: Numerical solution
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression

Linear Regression: Computational Complexity

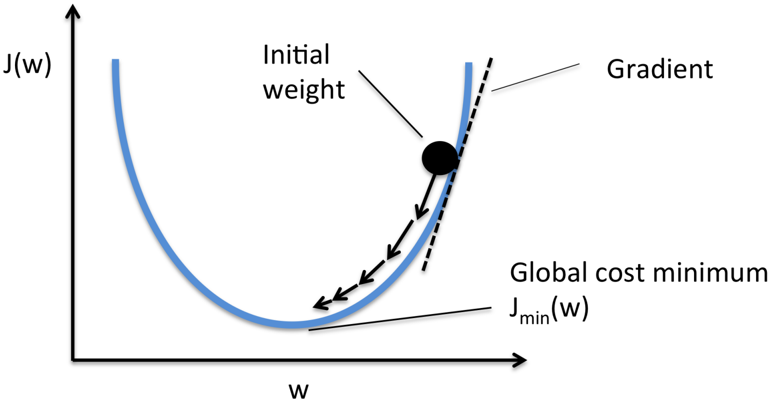
- Bottleneck for computing the solution $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ is to invert the matrix $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{D\times D}$
- Computational complexity is $O((D+1)^3)$ using Gauss-Jordan elimination
    - Impractical for large D
- Alternative
    - Find approximate solution through an iterative optimization algorithm
    - e.g. Gradient Descent

Gradient Descent

- Iterative algorithm finding a function's minimum via local search
- Standard optimization algorithm in many ML applications
  - e.g. linear regression, logistic regression
  - scales well to large datasets (e.g. no matrix multiplication)
  - proof that it solves many convex problems
  - good heuristic to non-convex problems as well
- **Input:** Function $J(\mathbf{w}) \in \mathbb{R}$
- **Output:** Local minimum $\mathbf{w}^*$
- **Goal:** Minimize $J(\mathbf{w})$ via greedy local search

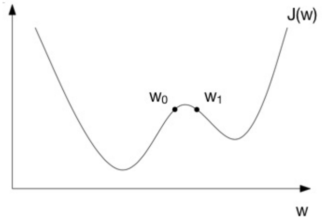
Gradient Descent

1-dimensional example: $J(w) = w^2$



Gradient Descent: 1-dimensional case

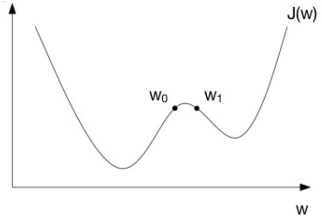
What will happen if we try to minimize $J(w)$ via a local search?



- Starting from w_0
  - We look to the right ($J(w) \uparrow$) and to the left ($J(w) \downarrow$)
  - We take a small step to the left
  - We repeat the above until we reach the left basin
- Starting from w_1
  - We similarly reach the right basin
- It is clear that the outcome depends on the starting point

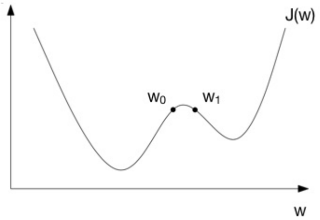
Gradient Descent: 1-dimensional case

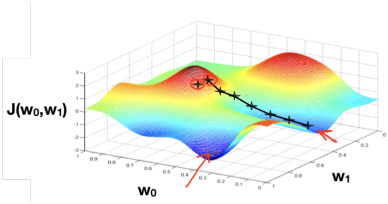
More formally, we do the following



- While $J'(w) \neq 0$
  - If $J'(w) > 0$ (i.e. $J(w) \uparrow$), move w a little bit to the left
  - If $J'(w) < 0$ (i.e. $J(w) \downarrow$), move w a little bit to the right
- The derivative $J'(w)$ is used to decide which direction to move
- In other words, move w towards the direction of $-J'(w)$

Gradient Descent: Algorithm Outline





[Source: Machine Learning, Coursera, Andrew Ng]

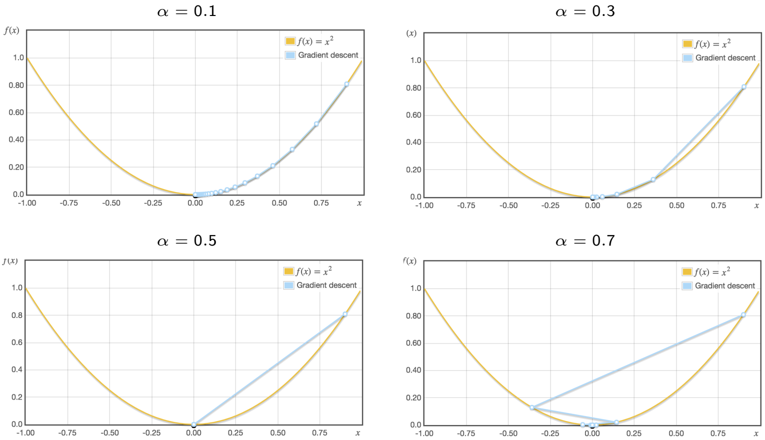
1-dimensional

1. Initialize w, ϵ, $\alpha(\cdot)$, $k := 0$
2. While $\left|\frac{dJ(w)}{dw}\right| > \epsilon$
   - 2a $k := k + 1$
   - 2b $w := w - \alpha(k) \cdot \frac{dJ(w)}{dw}$

d-dimensional

1. Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$
2. While $\|\nabla J(\mathbf{w})\|_2 > \epsilon$
   - 2a $k := k + 1$
   - 2b $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \nabla J(\mathbf{w})$

Gradient Descent: Step size (or learning rate) α



- If $\alpha(k)$ too small, convergence is unnecessarily slow
- If $\alpha(k)$ too large, correction process will overshoot and can diverge

Source: `http://www.onmyphd.com/?p=gradient.descent`

Gradient Descent: Step size (or learning rate) α

How to chose α?

- In practice, through experimentation
  - Check how $J(\mathbf{w})$ behaves over iterations for multiple α
  - α is a hyper-parameter
  - Therefore it can be tuned using a dev-set or a cross-validation framework

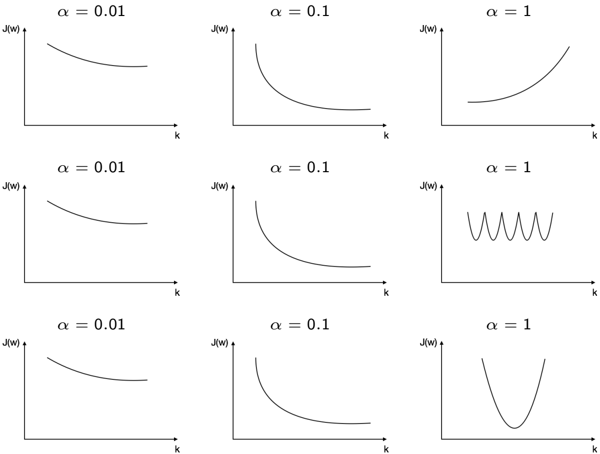
Gradient Descent: Step size (or learning rate) α

Question: A cost function $J(\mathbf{w})$ is optimized with Gradient Descent (GD) using different step size values α. We plot $J(\mathbf{w})$ w.r.t. the number of GD iterations k. Which of the following graph triplets can hold?

A

B

C

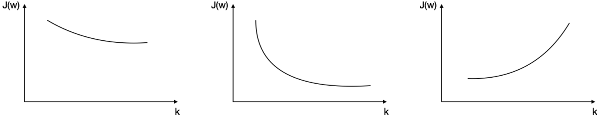


Gradient Descent: Step size (or learning rate) α

Question: A cost function $J(\mathbf{w})$ is optimized with Gradient Descent (GD) using different step size values α. We plot $J(\mathbf{w})$ w.r.t. the number of GD iterations k. Which of the following graph triplets can hold?

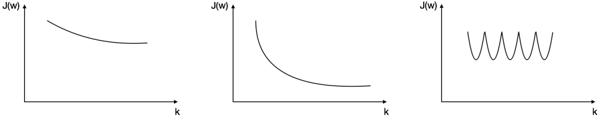
A

$\alpha = 0.01$ | $\alpha = 0.1$ | $\alpha = 1$



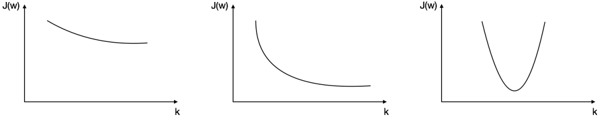
B

$\alpha = 0.01$ | $\alpha = 0.1$ | $\alpha = 1$



C

$\alpha = 0.01$ | $\alpha = 0.1$ | $\alpha = 1$



All answers can occur.

Gradient Descent: Stopping rule

- Hyper-parameter ϵ (i.e. $\|\nabla J(\mathbf{w})\|_2 > \epsilon$) determines when to stop
- Small ϵ: many iterations but higher quality solution
- Large ϵ: less iterations with the cost of more approximate solution
- How to chose ϵ in practice?
  - Try various values to achieve balance between cost and precision
  - Again use some type of cross-validation framework

- Hyperparameters: Parameters set before the beginning of the learning process (e.g. α, ϵ in gradient descent)
- Hyperparameter tuning: The process of choosing a set of optimal hyperparameters for the learning process
- Model parameters: The parameters learned during the learning process (e.g. weights $\mathbf{w}$ in linear regression)

Overview

- Linear Regression: What have we learnt so far
  - Representation
  - Analytic Solution: Ordinary Least Squares
- Linear Regression: Numerical solution
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression

Gradient Descent in Linear Regression

We can now derive the algorithm outline for minimizing the residual square sum (RSS) error of linear regression with gradient descent

- The residual sum of squares is the cost function:
$$\begin{aligned} J(\mathbf{w}) = RSS(\mathbf{w}) &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - 2(\mathbf{X}\mathbf{w})^T\mathbf{y} + (\mathbf{X}\mathbf{w})^T(\mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T(\mathbf{X}^T\mathbf{y}) + \mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} \end{aligned}$$
- Gradient Descent optimization expression:
$$\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \nabla J(\mathbf{w})$$
$$\nabla J(\mathbf{w}) = \frac{\vartheta RSS(\mathbf{w})}{\vartheta \mathbf{w}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w} = 0$$

Gradient Descent in Linear Regression

Question: Derive the algorithm outline for minimizing the residual square sum (RSS) error of linear regression with gradient descent

(Batch) Gradient Descent for Linear Regression

1 Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$

2 While $\|\nabla RSS(\mathbf{w})\|_2 > \epsilon$

  2a $k := k + 1$

  2b $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \left(\mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y}\right)$

Gradient Descent in Linear Regression

Stochastic Gradient Descent for Linear Regression

Update weights using one sample at a time

1. Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$
2. Loop until convergence
   - 2a $k := k + 1$
   - 2b Randomly choose a sample $(\mathbf{x_i}, y_i)$
   - 2c Compute its contribution to the gradient $\mathbf{g_i} = (\mathbf{x_i}^T \mathbf{w} - y_i) \cdot \mathbf{x_i}$
   - 2d Update the weights $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \mathbf{g_i}$

Gradient Descent in Linear Regression

Mini-Batch Gradient Descent for Linear Regression

Update weights using subset of samples at a time

1 Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$

2 Loop until convergence

   2a $k := k + 1$

   2b Randomly choose a subset of samples $S = \{(\mathbf{x_i}, y_i), \ldots, (\mathbf{x_{i+M}}, y_{i+M})\}$

   2c Form the mini-batch data matrix $\mathbf{X_S} = \begin{bmatrix} \mathbf{x_i}^T \\ \vdots \\ \mathbf{x}_{\mathbf{i+M}}^T \end{bmatrix}$

   2d Update the weights $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \left(\mathbf{X_S}^T \mathbf{X_S} \mathbf{w} - \mathbf{X_S}^T \mathbf{y} \right)$

- Good compromise between batch and stochastic gradient descent
- Common mini-batch sizes range between M=50-250 samples

Gradient Descent in Linear Regression

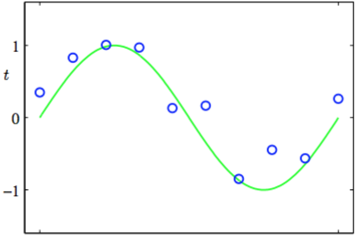
- Batch gradient descent computes exact gradient
- Stochastic gradient descent
  - Computes approximate gradient using one sample per iteration
  - Its expectation equals the true gradient
- Mini-batch gradient descent
  - Computes gradient based on subset of samples
- For large-scale problems stochastic or mini-batch descent often work well

Overview

- Linear Regression: What have we learnt so far
  - Representation
  - Analytic Solution: Ordinary Least Squares
- Linear Regression: Numerical solution
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression

What if data does not fit a line?

Example: Samples from a sine function



We can use a non-linear basis function
$\phi(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^D \rightarrow \mathbf{z} \in \mathbb{R}^M$

We can apply our linear regression model to the new features
$y_i = \mathbf{w}^T \mathbf{z_i} = \mathbf{w}^T \phi(\mathbf{x_i})$
$RSS(\mathbf{w}) = \sum_{n=1}^{N} \left(y_i - \mathbf{w}^T \phi(\mathbf{x_i})\right)^2, \ \mathbf{w} \in \mathbb{R}^M$
Example: $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$, $\phi(\mathbf{x}) = [x_1, x_2^2, x_1^3 + x_2]^T \in \mathbb{R}^3$

Non-Linear Basis Function

Residual sum of squares

$$RSS(\mathbf{w}) = \sum_{n=1}^{N} \left(y_i - \mathbf{w}^T \phi(\mathbf{x_i})\right)^2 = (\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})^T(\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})$$

Non-linear design matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(\mathbf{x_1})^T \\ \phi(\mathbf{x_2})^T \\ \vdots \\ \phi(\mathbf{x_N})^T \end{bmatrix} \in \mathbb{R}^{N \times M}$$

LMS solution with the non-linear design matrix

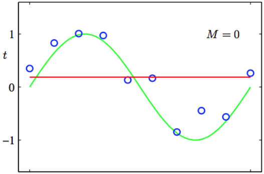
$$\mathbf{w}^{LMS} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$$

Non-Linear Basis Function

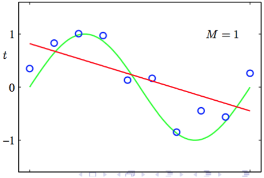
Example: Samples from a sine function

Polynomial basis function $\phi(\mathbf{x}) = [1\ x\ \ldots\ x^M]^T$

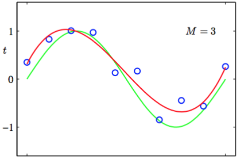
$M = 0$ underfitting



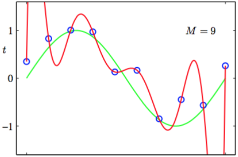
$M = 1$ underfitting



$M = 3$



$M = 9$ overfitting



Overfitting

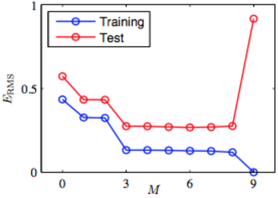
Weights of high order polynomials are very large

$$y_i = \mathbf{w}^T \mathbf{z_i} = \mathbf{w}^T \phi(\mathbf{x_i}), \quad \mathbf{z_i} = \phi(\mathbf{x_i}) \in \mathbb{R}^M$$

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Overfitting

- The risk of using highly flexible (complicated) models without enough data
- Leads to poor generalization
- How to detect overfitting?
  - Plot model complexity (e.g. polynomial order) versus objective function
  - As complexity increases, performance on training improves, while on testing first improves and then deteriorates
- How to avoid overfitting?
  - More data or regularization



Linear Regression: To summarize

- **Representation:** linear and non-linear basis

$$f: \mathbf{x} \to y, \quad f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$
$$f: \mathbf{z} \to y, \quad f(\mathbf{x}) = \mathbf{w}^T\mathbf{z} = \mathbf{w}^T\phi(\mathbf{x}), \quad \phi: \mathbf{x} \in \mathbb{R}^D \to \mathbf{z} \in \mathbb{R}^M$$

- **Evaluation**: Minimizing residual sum of squares

$$\min_{\mathbf{w}} RSS(\mathbf{w}), \quad RSS(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$
$$\min_{\mathbf{w}} RSS(\mathbf{w}), \quad RSS(\mathbf{w}) = (\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})^T(\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})$$

- **Analytic Optimization**: Ordinary least squares (OLS) solution

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \quad \mathbf{w}^* = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$$

- **Approximate Optimization**: Gradient descent
- **Readings:** Alpaydin Ch 2, Abu-Mostafa Ch 3.2; Handouts 6,7 on Piazza
- **Code example:** *GradientDescentExample.ipynb* on Google Drive
`https://colab.research.google.com/drive/1msq7fZ4Z88MIr6qaJStfEgsYMQ40sy3R`